#SAT алгоритм для малых PTF-схем константной глубины

Лялина Альбина

Современные методы в теоретической информатике

26 марта 2019 г.

Введение

Определение 2

#SAT для k-PTF
Вход: k-PTF f, заданная своим полиномом P с целыми коэффициентами.
Выход: Число выполняющих наборов f (число таких a, что $P(a) < 0$).

Теорема 4

Пусть k - фиксированная константа. Тогда существует вероятностный алгоритм с нулевой ошибкой, который решает #SAT для k-PTF за время $poly(n, M)2^{n-s}$, где $s = \tilde{\Omega}(n^{\frac{1}{k+1}})$.

Введение

Определение 6

#SAT для k-PTF схем
Вход: k-PTF схема C, заданная своим набором полиномов P с целыми коэффициентами.
Выход: Число выполняющих наборов C.

Теорема 7

Пусть k и d - фиксированные константы. Тогда для некоторого $\varepsilon_{k,d} > 0$ существует вероятностный алгоритм с нулевой ошибкой, который решает #SAT для k-PTF схем размера $\leq s = n^{1+\varepsilon_{k,d}}$ с вероятностью $\geq 1/4$ и выводит ? в противном случае. Алгоритм работает за время $poly(n, M)2^{n-z}$, где $z = n^{\varepsilon_{k,d}}$.

Результат из [KLMZ17]

Теорема 12

Пусть k и c -фиксированные константы. Пусть $r = \sum_{i=0}^{k} \binom{m}{i} = \Theta(m^k)$ обозначает число коэффициентов в многочлене kой степени от m переменных. Тогда существует вероятностный алгоритм $\mathcal{A}_{12}$, работающий за время $2^{O(\Delta)}$, который по $m, l \in \mathbb{Z}_+$, где $l \leq m^c$, выдает случайное LDT T глубины $\Delta = O(l \cdot m^{k+1} \log m)$, которое вычисляет случайную функцию $F : \mathbb{R}^{r \cdot l} \to \mathbb{N} \cup \{?\}$ со свойством:
Для каждого $w = (coeff_{m,k}(P_1), \ldots, coeff_{m,k}(P_l)) \in \mathbb{R}^{r \cdot l}$ (где $\forall a \in \{-1, 1\}^m \, \mathrm{P}_i(a) \neq 0$), $F(w)$ это либо число общих выполняющих наборов для всех k-PTF, представимых с помощью $P_1, \ldots, P_l$, либо равно ?.

Алгоритм $\mathcal{A}$ для PTF-SAT (Доказательство теоремы 4)

Вход: $P \in \mathbb{Z}[x_1, \ldots, x_n]$ степени не больше k, параметры (n, M), также завиксируем, что $m = n^{\frac{1}{k+1}} / \log n$.

Алгоритм $\mathcal{A}$:

1. Сделать $n_1 = 10n$ итераций алгоритма $\mathcal{A}_{12}$ для $l = 1$ (число полиномов), тем самым сконструировав LDT $T_1, \ldots, T_{n_1}$. Каждое T_i выдает число выполняющих наборов для произвольной k-PTF g от m переменных с вероятностью $> \frac{1}{2}$.
2. $N = 0$ *% будет числом выполняющих наборов*
3. Для каждого означивания $\alpha \in \{-1, 1\}^{n-m}$ для $x_{m+1}, \ldots, x_n$ вычисляем P_α полученный из P соответствующим означиванием. Прогоняем все $T_1, \ldots, T_{n_1}$ на P_α. Если все выдали ?, то выдаем ?, если кто-то выдал N_α, то $N+ = N_\alpha$.
4. Вывести N.

Алгоритм для PTF схем. Процедура $\mathcal{A}_1(n', s, f_1, \ldots, f_s)$

Вход: AND от k-PTF $f_1, \ldots, f_s$, $s \leq n^{0.1}$, f_i зависит от $n' \leq n^{1/(2(k+1))}$ переменных.
Вывод: Число общих удовлетворяющих наборов для f_i.

Идея: Возьмем $n_1 = 10n$, и прогоняем $\mathcal{A}_{12}$ для $l = n^{0.1}$ и $m = n^{1/(2(k+1))}$ n_1 раз.Соответствующие деревья и будут каждое решать нашу задачу с вероятностью ошибки $1/2$.

Время на создание конструкции: $2^{O(n^{0.6})}$.
Время работы: $poly(n, M)$.
Вероятность ошибки: $1/2^{10n}$.

Алгоритм для PTF схем. Процедура $\mathcal{A}_2(C, d, n, M)$

Пусть $1 > \varepsilon >> (\log n)^{-1}, \beta = A\varepsilon, \delta = exp(-n^{\beta/B \cdot k^2})$, где A и B очень большие константы, k степень многочленов.

Вход: k-PTF схема C глубины d от n переменных размера $n^{1+\varepsilon}$ и веса M.

Вывод: Дерево поиска T_{DT} глубины $n - n^{1-2\beta}$, такое, что для равномерно выбранного листа $\sigma \in \{-1, 1\}^{n-n^{1-2\beta}}$, оно выводит *хорошую* схему с вероятностью $1 - exp(-n^{\varepsilon})$. *Хорошей* называется схема, для которой верно:

- Не более чем n гейтов δ-близки к константе.
- Не более чем n^{β} гейтов не δ-близки к константе.

Идея: см. [KL18].

Время работы: $poly(n, M) \cdot O(2^{n-n^{1-2\beta}})$.

Вероятность ошибки: $1/2^{10n}$.

Алгоритм для PTF схем. Процедура $\mathcal{A}_3(m, l, \delta, g_1, \ldots, g_l)$

Этот алгоритм **детерменированный**!

Вход: Параметры $m \leq n, l, \delta$ такие, что $\delta \in [exp(-m^{1/10(k+1)}), 1]$, $l \leq m^2$, k-PTF $g_1, \ldots, g_l$, заданные полиномами $P_1, \ldots, P_l$ от m переменных $x_1, \ldots, x_m$ с весом M, каждый из которых δ-близок к -1.

Вывод: Все такие $a \in \{-1, 1\}^m$, что $P_i(a) > 0$ для некоторого i.
Оракулы: $\mathcal{A}_1$.

Идея: Пусть $q = \frac{1}{2} \log \frac{1}{\delta} < \frac{m}{2}$. Для каждого частичного означивания применяем $x_{q+1}, \ldots, x_m$ решить задачу с помощью $\mathcal{A}_1$ (выводит 2^m если всегда -1, такие дальше нечего и проверять, в противном случае дальше перебором 2^q вариантов).

Время работы: $poly(n, M) \cdot \sqrt{\delta} \cdot 2^m$.

Алгоритм для PTF схем. Процедура $\mathcal{A}_4(n, M, g_1, \ldots, g_\tau)$

Вход: Набор k-PTF $g_1, \ldots, g_\tau$, заданный полиномами $P_1, \ldots, P_\tau$ от n переменных с весом не больше M и условием $\sum_{i=1}^{\tau} fanin(g_i) \leq n^{1+\varepsilon_1}$, где $\varepsilon_1 = 1/10A$.
Вывод: Число наборов $a \in \{-1, 1\}^n$ таких, что $P_i(a) < 0$ для любого i.
Оракулы: $\mathcal{A}_1$

Идея: Использовать алгоритм $\mathcal{A}_2(C, 2, n, M)$ для создания T_{DT}. Для листьев схем не являющихся *хорошими* делать просто перебором, для хороших использовать $\mathcal{A}_1$.

Время работы: $poly(n, M) \cdot 2^{n-n^\alpha}$, где $\alpha = O(\varepsilon_1/(k+1))$.
Вероятность ошибки: $1/2$.

Алгоритм для PTF схем. $\mathcal{A}_5(n, d, M, n^{1+\varepsilon_d}, C, \mathcal{P})$

Вход: Набор k-PTF $\mathcal{P} = g_1, \ldots, g_\tau$ от n переменных с весом не больше M и условием $\sum_{i=1}^{\tau} fanin(g_i) \leq n^{1+\varepsilon_d}$ и k-PTF схема C от n переменных глубины d размера $n^{1+\varepsilon_d}$ и веса M.

Вывод: Число общих выполняющих наборов для C и $\mathcal{P}$, то есть число $a \in \{-1, 1\}^n$ таких, что $P_i(a) < 0$ для любого i и $C(a) < 0$.

Оракулы: $\mathcal{A}_1$ и $\mathcal{A}_4$

Идея: Рекурсивно по глубине схемы d. Если глубина 1, то пользуемся $\mathcal{A}_4$. В противном случае строим T_{DT}:

для не хороших схем перебор.

для хороших схем:

- для δ-близких к константе гейтов нижнего уровня:
  - для минорных означиваний, находим их с помощью $\mathcal{A}_3$, и проверяем.
  - для не минорных означиваний, склоняемся к соответствующей константе.
- для не δ-близких к константе гейтов нижнего уровня:
  - угадываем их и применяемся рекурсивно.

Время работы: $poly(n, M) \cdot 2^{n - n^{O(\varepsilon_d/(k+1))}}$.

Вероятность ошибки: $1/2$.

Алгоритм для PTF схем. $\mathcal{B}(C, n, d, M, n^{1+\varepsilon_d}, k)$

Вход: k-PTF схема C от n переменных глубины d размера $n^{1+\varepsilon_d}$ и веса M.
Вывод: Число выполняющих наборов для C , то есть число $a \in \{-1, 1\}^n$ таких, что $C(a) < 0$.

Идея: Это просто запуск $\mathcal{A}_5(n, d, M, n^{1+\varepsilon_d}, C, \emptyset)$ и симуляция оракула $\mathcal{A}_1$.

Время работы: $2^{O(n^{0.6})} + poly(n, M) \cdot 2^{n - n^{O(\varepsilon_d/(k+1))}}$.
Вероятность ошибки: $1/4$.

The End